AUTOMATING UVM BASED FUNCTIONAL AND DESIGN VERIFICATION FOR SOC USING INTELLIGENT ALGORITHM

A SYNOPSIS

Submitted by

BHUVANESWARY N

(Reg. No. 201808264)

in partial fulfillment for the award of the degree

of

DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

SCHOOL OF ELECTRONICS ELECTRICAL AND BIOMEDICAL TECHNOLOGY

KALASALINGAM ACADEMY OF RESEARCH AND EDUCATION

(DEEMED TO BE UNIVERSITY)

ANAND NAGAR, KRISHNANKOIL-626126

TAMILNADU, INDIA.

JUNE 2023

1. INTRODUCTION

Testing has been an important process of circuit framework because of the expanding complexities of embedded design systems. Conventional techniques for testing and troubleshooting require the utilization of logic analyzers and in-circuit emulators. Be that as it may, other than the excessive cost of the devices, little, reduced, and incorporated frameworks would make testing exceedingly troublesome or indeed, even inconceivable. A well-known and financial test technique for getting to modules in incorporated circuit outlines and processors is the usage of the IEEE 1149.1-1990 that is incorporation of verification methodology standard. The boundary scan design of the incorporation of verification methodology standard can give test support works that can be utilized to impart, load and test guidelines, information, and control signs to and from the device under test known as DUT. Several testing algorithms has been used for the different devices testing and some of the advanced test features such as Built-In Self-Test(BIST), Boundary scan (incorporation of verification methodology), Automatic test Pattern generation(ATPG) etc. But the consumption of time, intelligent frame for accuracy measurements are still in the darker side.

In every methodology reusability is considered as significant issues which challenges the developers. We can say, methodology is set of libraries utilized to build test benches. There various methodologies are available for the verification of SoC. When compared to other traditional methods like OVM, presently the UVM (“Universal Verification Methodology”) plays vital role in testing lot of benchmarks [1,2]. This kind of design verification incorporates more random simulations with an extensive set of user-controlled combinations, allowing for a rich and diversified generation of the input's sequences. However, it is time-consuming and insufficient to fully cover the complete design functionalities [3].

Recently, Machine Learning based verification methodologies are gaining the brighter light of research for an automatic and efficient verification process. But still, improvisation is required in terms of time and computational complexity. This research proposed the novel hybrid reinforced feedforward networks for better coverage ratio and less time complexity to overcome this problem.

Section 2 explains the literature review, Section 3 explains the objectives, and then the research problem is identified in Section 4, research contribution is presented in Section 5, thesis organization is presented in Section 6, the proposed methodology is presented in Section 7 and

result of the proposed work is explained in Section 8. This proposed work is concluded with Performance analysis in Section 9.

2. LITERATURE SURVEY

Tahera K et al. [4] studied complex testing methodologies, especially in the automation industry. The challenges of creating complex products in designing, manufacturing, and validating the software are still crucial. There are many significant vital factors where the developer should focus on product design as per the application requirement. Before the designing process is initiated, the VLSI developer should verify the hardware/software availability, materials availability, and testing tools for each product development. Various scientists have considered the criticalities involved in such a process separately in recent days. Similarly, Mdhammadabdul Naveed et al. [5] studied various testing methodologies involved in SoC development, especially chips involved in computational applications. The author addressed the basic testing flow for VLSI inbuilt ICs and a few recent models engaged in performance improvement. The authors mainly optimized the circuit design by minimizing the number of ALU blocks in the VLSI circuit. According to the Vidhya Lakshmi et al. [6] power scattering has become a primary boundary in low force VLSI circuit designs due to the widespread use of tiny electronic devices and the appraisal of microelectronic advancement. The circuit complexity and speed of rising VLSI innovation predict a significant rise in force utilization. The vitality scattering in low force CMOS VLSI circuits is caused by charging and discharging of internal hub capacitances due to progress movement, which is one of the primary points that also influences the dynamic force dissemination. The reduction in force, territory, and speed necessitates simplification at all levels of design techniques. Konstantin Smirnov et al. [7] analyzed the "CAD Functional Test-Studio" VLSI testing tool. VLSI verification for failure-free performance while switching control strategies is provided. The complex's numerical method lets you pinpoint critical components in the VLSI layout. The results of functional VLSI control have been stored in a uniform data format. The format is immune to failures of electronic instruments. Chenchen Xie et al. [8] proposed the "Built-in-self-test (BIST)-phase memory model (PCM)" memory-based algorithm for detecting the faults in VLSI circuits to minimize the error rate during production. An enhanced march test algorithm is proposed to achieve 100 percent fault coverage and diagnostic accuracy in bit-oriented PCM by focusing on specific PCM faults and others. After that, the proposed algorithm is translated to word-oriented PCM and given the capacity to detect probable intraword impact. In

addition, a unique defect information storage scheme is designed to lessen the memory test's reliance on external devices. This strategy has been shown to improve the chance of discovering the predetermined fault-free zones in the tested memory using modeling and simulation in C. According to Sushanth Varada et al. [9] the Viterbi Algorithm is a recursive ideal response for calculating the most likely state clustering of discrete-time finite-state Markov measure and Hidden Markov Models (HMM) seen in deterministic chaos. In the restriction length k, the Viterbi computation is commonly used for deciphering modulation schemes, including its use in digital correspondences explicitly in satellite and cell interchanges. The Viterbi computation or modifications are used in capacity devices to expedite accessibility, discourse combination, and recognition improvements. Uthayakumar et al. [10] proposed a "Duty Approximate Testing framework" is introduced that generates modulation schemes for only separate faults. The equalization strategy eliminates flaws, and filtering is used in the proposed DPR technique for faults. The proposed DPR is frequently used to reduce the amount of space available for running storage. We discovered that our proposed method gives an unlimited output speed on the design deployment, reducing completion time to a few minutes, using experimental investigation. From the literature survey it is identified that there is complexity in implementation, design time and computation time is too high, user friendly is missing and hence require more enhancement.

3. OBJECTIVES OF THE WORK

The main objectives of this research work are

- To design the ARM based Verification Methodology for less computation in verification methodology for different architectures
- To implement the System –Python for Verification methodology to increase the features of UVM.
- To test the various IP/SoC architectures using the System-Python based Verification methodology.

4. RESEARCH PROBLEM

Generally, the major complex part of circuit designs is System and Functional verification because of its high time consumption. So, it is highly essential to minimize the verification time. To overcome those mentioned above constraints, complex circuits like "ASICs and System on

chips (SoC)" are increasing significantly. [11, 12] Universal Verification Methodology (UVM) occupies the most predominant role in the verification mechanisms. UVM consists of the UVC sequencer, monitor, drivers, and scoreboards. The UVC Sequencer creates Sequences and passes them to a Driver, which organizes the data and sends them to the DUT.

Meanwhile, a Monitor collects data sent and forwards it to the scoreboard for checking. Another UVC collects the output data through a Monitor and feeds the same scoreboard. This kind of design verification incorporates more random simulations with an extensive set of user-controlled combinations, allowing for a rich and diversified generation of the input's sequences. However, it is time-consuming and insufficient to fully cover the complete design functionalities.

5. CONTRIBUTION

1. First, the development of high speed and high coverage rate design verification (DV) tool for handling the low-level circuits to Multiple Complex SoC designs
2. Second, A novel dataset generation to be used for the deep learning-based DV Systems
3. Third, intelligent deep learning-based design verification (DV) tool needs to be designed for the automatic and high accurate verification tool.
4. Finally, optimization of input sequences for achieving high speed and less time complexity.

6. ORGANIZATION OF THESIS

The research work is organized into four chapters and is as follows:

Chapter 1 - Introduction

Chapter 2 – Deals with literature survey

Chapter 3 – Deals with design and implementation of intelligent ARM based verification methodology for low computation time and high performance

Chapter 4 – Discussed exploration on reusability of Universal Verification Methodology

Chapter 5 – Discussed an optimized hybrid based deep reinforcement neural network for verification methodology.

Chapter 6 – Concludes the research methodology with future aspects.

7. PROPOSED METHODOLOGY

The proposed methodology has 4 phases and they are explained as follows

7.1 AVM (ARM BASED VERIFICATION METHODOLOGY)

UVM s are used for verification, separate usage of the different tests for the devices are carried out manually and automatically which consumes more time, and more overhead. Hence high speed Testing and integration of the tools are the most challenging part and still remains in the darker side of the research. With this, a new intelligent configuration tool called AVM (ARM BASED VERIFICATION METHODOLOGY) methodology has been proposed [13].

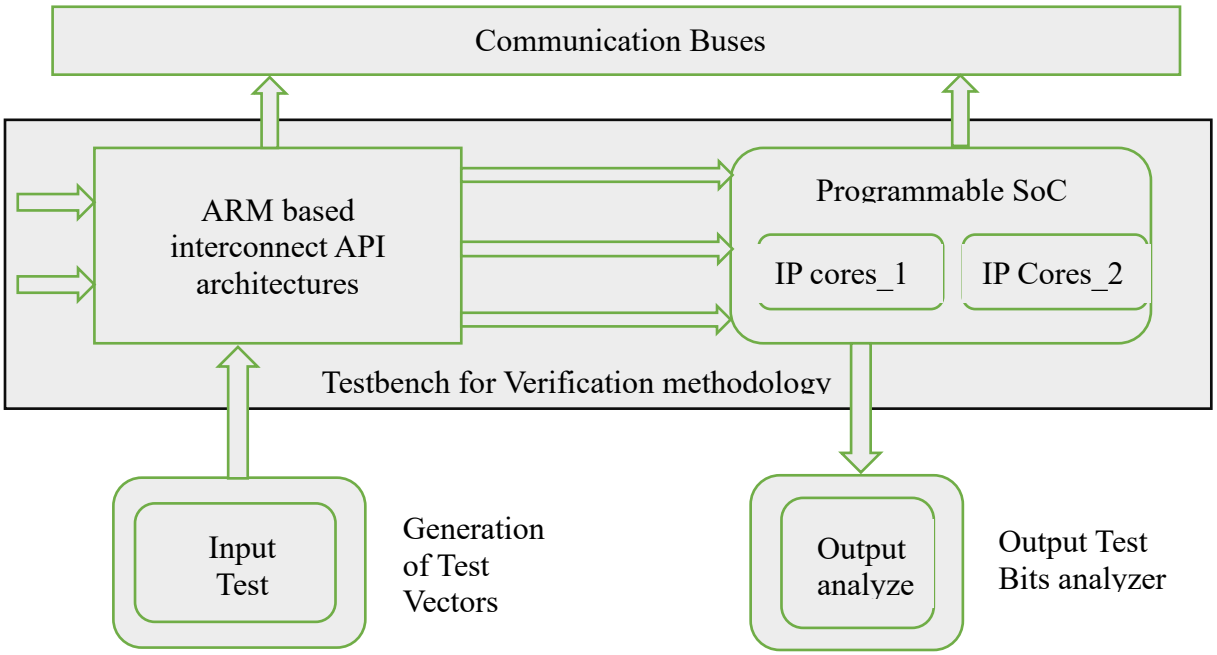

Figure 7.1 Architecture of AVM

Figure 7.1 shows the AVM architecture. The intelligent configuration software has been developed on Embedded ARM architectures interfaced with the Programmable SOCs. The proposed tool AVM consists of two important stages such as the Pre-Processing and Intelligent Post-Processing phases in which the output test vectors are analyzed on the principles of Rule based framework [14]. AVM is multi layered Architecture with the specification of the reconfigurable Clock speed adjustments in accordance with the test vectors. The working methodology is categorized into four different modes of operations which are represented in Table 7.1.

Table 7.1: Modes of Operation in AVM Configuration Software

S.NO	MODES OF OPERATIONS	FUNCTIOING MODES
01	OP_MODE_1	Scanning Modes
02	OP_MODE_2	Functional Test Modes
03	OP_MODE_3	Functional Test Mechanisms with the test vectors
04	OP_MODE_4	Analyzing Modes.

The test vectors are produced relying upon the directions of the client. The directions choose the ID which is referred to be as identification proof vector which constrains the generation of the test vectors and it is focused for the rapid testing. The ARM design has coordinated with the product test vector generator, which is nothing but, the type of the Pseudo random generator) controlled by programmable control lines which work on clock speed selected by the user which are shown in Figure 7.2.

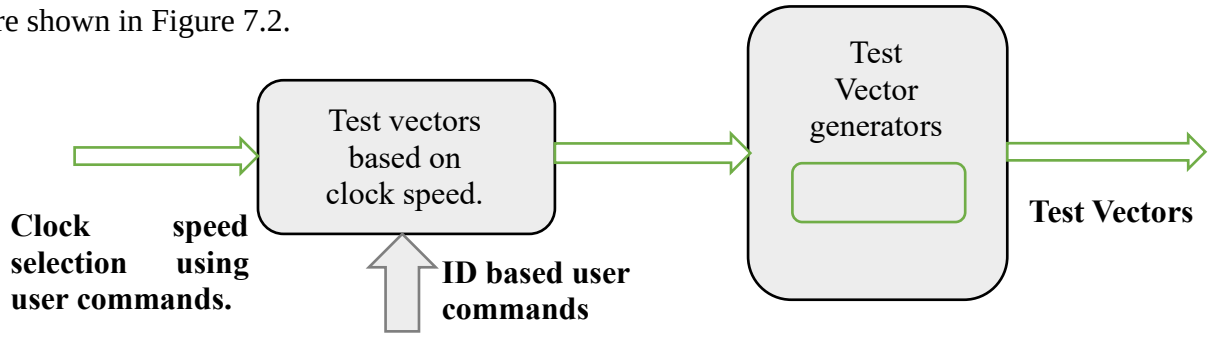
Test vectors based on clock speed.
Test Vector generators
Clock speed selection using user commands.
ID based user commands
Test Vectors

Figure 7.2 Software Test Vector Generators with PCL- Programmable Control Lines.

The proposed test bench AVM combines this feature of switching from the Control to Functional testing mode which is extremely used for the low power testing environment. Hence proposed methodology is suitable for the low –level SOC and once features increases, complexity and time increases linearly [15].

7.2 HYBRID LEARNING BASED PYTHON INTEGRATED VERIFICATION ENVIRONMENT FOR SOC VERIFICATION METHODOLOGIES

PIVE framework mechanisms are packages, and it tends to be powerfully customized by the client [16]. The primary parts of PIVE are "Source generators, Driver screen, catch, Diagnosis board, natural factors and client characterized tests".

Scheduler and API Modeler: The Scheduler and Application Programmable Interfaces provides the scheduling of the test stimuli to the DUV and interfacing from the entities to the IP is done by Application Programmable Modeling layer. In order to “reduce the time period” and increase the “accuracy”, stimulus are changed over to packets and sent to the verification IP with the distinctive time allotments”. The Skeleton of the proposed tool is displayed in Figure 3.

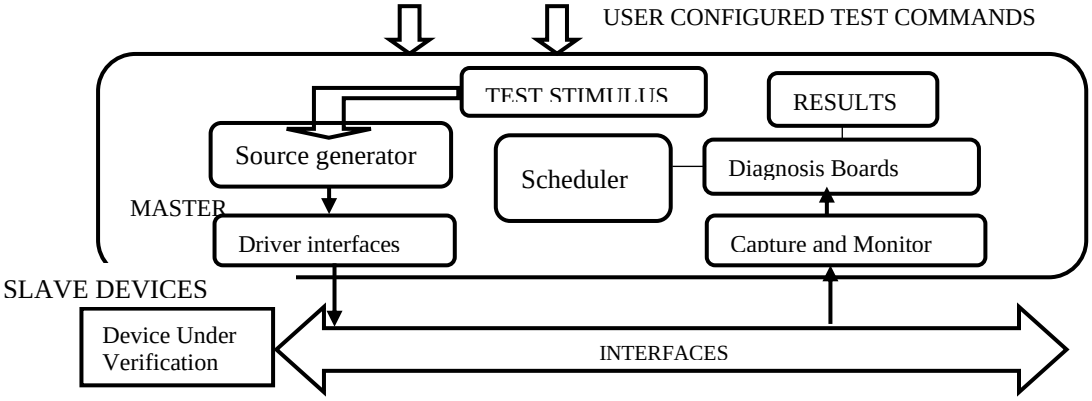

Figure 7.3 Overall Skeleton of PIVE

Figure 7.3 overall skeleton of PIVE Tools Master Source generator generates the data depends upon the user’ commands and it is sent through the DUV by the driver. The data are received by the slave (DUV) and sent to the diagnosis board where the comparison between the received data and transmitted data take place. The components are reused for different applications.

IP based PIVE Environment verification steps:

The mandatory steps of PIVE verification methodology is presented as follows

Step 1: Complete DUT Implementation

Step 2: Preparation of Verification Plan and Reference Model

Step 3: Verification Environment Building the like "Interface, Diagnosis Board, screens, experiments, and Simulation".

Step 4: Utilizing python packages to Run the Regressions test and to do comparative analysis as per the reference plan.

7.3 EXPLORATION ON REUSABILITY OF UNIVERSAL VERIFICATION METHODOLOGY

With the usage of System Python, new algorithm of verification Hybrid Universal Methodology (HUM) has been proposed. It is an Integration of the Fault Diagnosis System along with Verification methodology. It uses the System Python as UVM enhanced feature for the verification [16,17]. The Figure 7.4 shows the complete structure of Universal Verification Methodology

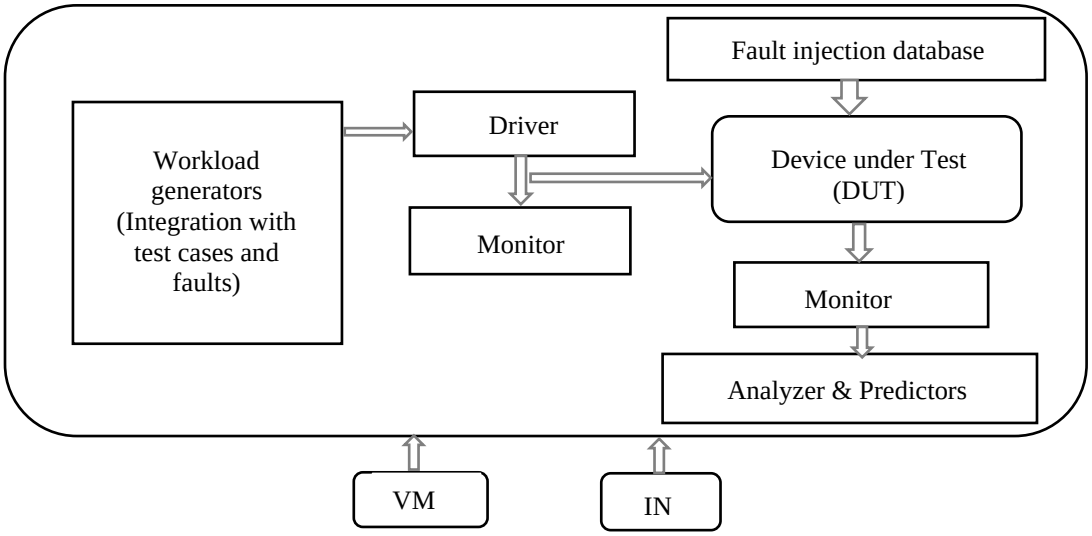

Figure 7.4 Complete Structure of Universal Verification Methodology

Workload Generators: These are the entities that are used to generate the stimulus codes and send the sequence codes to the driver. The Source generators are controlled by the user defined test which sits on the top layer of the entity. The workload generators are similar to the sequence generators in UVM where this module is used for the generation of the sequences for the verifications and different faults sequences in the fault injection module.

Drivers: The drivers or does the two important functionality one is to send the stimulus in accordance with the Device under verification (DUV) interface and to monitor the device traffic between the testbench and DUV. The drivers always reside downside to the source generators and take care about level conversion in accordance with the device under verification. Stimulus are

converted to packets and sent to the verification IP with the different time slots which can increase in accuracy and reduction in time period.

Monitor and Capture: The monitor and capture are used for capturing the traffics, word length between the drivers, DUV and stores it in the further classification.

Communication Interface: Communication Interfaces play the role of TLM in the UVM which provides the communication interfaces between the drivers and DUV.

Predictors: The predictors are used in replacing the scoreboards. The predictors consists of the different modules such as the data storage from the device under test(DUT) , and intelligent data analyzer. The outputs from the DUT will be stored in the databases similar to the UVM as uvm_config_db. The predictors compare the inputs from the DUT to the Golden Model /reference model stored in the databases.

Programming Models: Normally the API for the UVM has been developed using Accerella System C/ System Verilog programming and scoreboards/predictors uses C/C++ direct programming on the UVM components. The proposed methodology uses the Python for the both cases, such as the System Python is used for the API Interfaces and Python is used for developing the predictors for the comparing the test outputs.

7.3 AN OPTIMIZED HYBRID BASED DEEP REINFORCEMENT NEURAL NETWORK FOR VERIFICATION METHODOLOGY

Figure 7.5 shows the design verification framework. The DV framework consists of simulated random sequence generators, the device under test, analyzers, a deep learning network, a scoreboard, and an optimized random sequencer. The flow of the proposed framework is as follows: random sequence generators are used to generate the sequences for verifying the outputs for each path of circuits. The collected outputs are stored in analyzers and fed to the learning framework. The proposed trained deep learning model has two functions: 1) Analyzing the Outputs obtained from the different paths of the circuits and 2) To identify the sequence of uncovered vectors to attain the target states. In a traditional verification environment, the complexity of identifying all the output paths increases as the design complexity increases. The proposed framework could be a better solution by providing path sequences to verify all the paths accurately. Finally, the verified output paths will be re-coded on the scoreboard.

A number of tests for devices are tested with random stimulus. For each input to the Device Under Test (DUT), outputs from the test, coverage information and status are captured. From the collected data, important features such as switching probability, number of logic gates present in the path with corresponding inputs and outputs are calculated. These features are used for training the proposed Deep learning architecture [18].

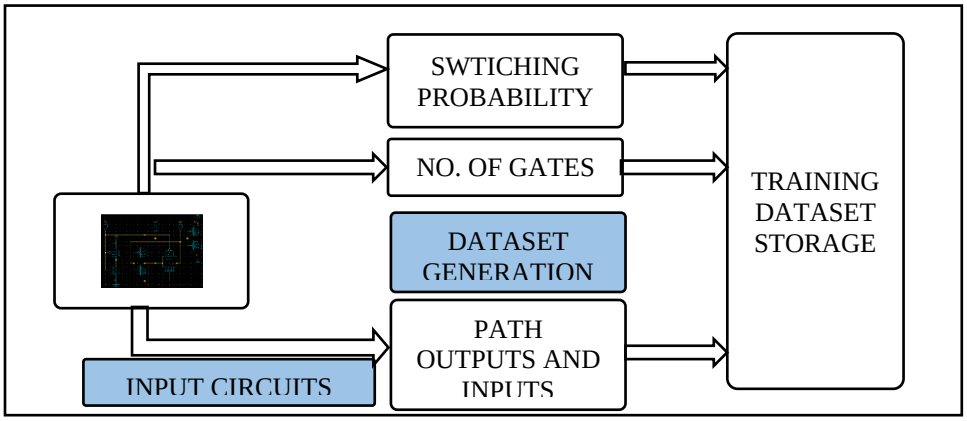

Figure 7.5 Overall Framework for the Proposed Design Verification Tool

- **DATASET GENERATION**

In this work, datasets are collected based on the randomization of outputs for randomized inputs. These data can't be managed using any "direct mapping" but only by the logic function and circuits present in the complex designs. Hence, the proposed architecture incorporates the three essential features: switching probability and the number of logic gates present in the path with corresponding inputs and outputs.

- **SWITCHING PROBABILITY (SP)**

For an efficient dataset collection, design circuits are converted into distributed graphs in which the nodes in the graphs represent the circuits' gates. SP of the nodes represents the probability of the nodes switching ranging from 0 to 1. Output Switching signals are essential functions that map strongly with input signal behavior. The mathematical expression for output probabilities can be represented as $(P)= P(A=0)+P(A=1)$ (1)

Where P(A=0) represents the probability of output transitions at input=0 and P(A=1) represents the probability of output transitions at input =1.

SP of a node relies on "input probabilities and logic function" from input to that node. Once the output value is known, we can quickly identify which output value is affected by the input. The output SP is an essential consideration for output randomization and test-case failure

because of input randomization. Since the switching probability depends on the number of nodes in the circuits, collecting the number of gates present in the nodes is mandatory.
The switching probability of each gate is mentioned in Table 7.2

Table 7.2 Switching Probabilities of different Gate Structure

S.No	Gates	Switching probability
1	NAND	$\boldsymbol{P(NAND) = P(o) * (1 - P(o))}$
2	NOR	$\boldsymbol{P((NOR) = P(o) * (1 + P(o))}$
3	AND	$\boldsymbol{P(AND) = P(o) * P(I)}$
4	OR	$\boldsymbol{P(OR) = P(o) * P(I)}$

- **OUTPUT AND INPUT PATHS**

Finally, the inputs and outputs from the nodes in the path are calculated and fed as the training vectors. Three input features, such as switching probabilities number of gates with their corresponding the inputs and outputs, are measured, and these are considered as the input to the deep learning algorithms

- **RL-DNN BASED VERIFICATION TOOL**

The deep learning-based framework has been adopted for the verification mechanism. The Deep Extreme learning Engine and RL framework run parallel in the proposed framework. All the inputs and outputs are grouped as mentioned in Algorithm-1. The proposed system number of DELM networks relies on the assertions of the circuits. DELM is trained based on the input data and labeled based on the characteristics of the assertions [19].

8. RESULTS AND DISCUSSION

8.1 RESULTS ON AVPM

At Stuck-at –Faults Diagnosis and Time Detection

From Table 8.1, it is clearly illustrated that the time is readily reduced by the adoption of the Programmable Clock Speeds and the Software Controlled random generators integrated in AVM and accuracy for different test vectors represented in Table 8.2.

Table 8.1 Different Time Taken for the ITC'02 SOC benchmarks circuits implemented in SOC for testing with the single user defined input vectors and Diagnosis at Programmable Frequency of 12MHZ, 60 MHz, 600 MHz, 1.2 MHz

S. No	Bench Mark SOC	Execution time (s) at 12 MHz	Execution time (s) at 60 MHz	Execution time (s) at 600 MHz	Execution time (s) at 1.2 MHz
1	A586710	201.189	191.189	101.189	91.189
2	D281	191.56	161.56	91.56	83.56
3	D695	300.245	270.245	150.245	90.245
4	F2126	233.67	213.67	115.67	93.67
5	G1023	231.90	221.90	131.90	81.90
6	H953	246.76	226.76	126.76	86.76
7	P22810	255.30	235.30	125.30	85.30
8	P34392	264.50	164.50	164.50	84.50
9	P93791	266.89	166.89	166.89	86.89
10	Q12710	244.45	144.45	124.45	84.45
11	T512505	215.6	185.6	105.6	84.45

Table 8.2: Represents the Accuracy Measurement for the Different test vectors under 600 – 120000 MHz

S.NO	Bench Mark Circuits	No of test Vectors	No of Faults Detected	Diagnosis Rate(%)
01	A586710	100	98	98.0
02	D281	101	97	96.03
03	D695	102	98	96.07
04	F2126	105	98	93.33
05	G1023	115	110	95.63
06	H953	123	120	97.50
07	P22810	122	120	97.50

08	P34392	123	120	97.50
09	P93791	122	119	96.45
10	Q12710	122	119	96.45
11	T512505	122	119	96.45

Moreover, the Proposed test bench has been compared with the other algorithms proposed by Khaled Fathy and Ammun Gruwell [3] in Table 8.3 given below. It clearly states that reduction in time, accuracy and other features are outperformed by the proposed algorithms.

Table 8.3 Represents the Comparative Analysis of the AVM with other Algorithm

S.No	**Algorithm Used**	**Accuracy Obtained**	**Time in reduction**	**No of Circuits tested**	**Features Incorporated**
01	AVM	95-96%	Low	>800	High

8.2 RESULTS OF PIVE

Table 8.4 Represents the Comparative Analysis: System Verilog based UVM Vs PIVE

SoCs	Parameters	System Verilog based UVM	PIVE(Python Based Programming)
STORM SOC	Regression Time	0.35 H	0.15 H
	Development	15 H (test bench design time)	06 H (test bench design time)
OPENSPARC SOC	Regression Time	0.47 H	0.20 H
	Development	18 H (test bench design time)	07 H (test bench design time)
GECK03 SOC	Regression Time	1.45 H	0.45 H
	Development	10 H (test bench design time)	05 H (test bench design time)
WISHBONE SOC	Regression Time	0.25 H	0.15 H
	Development	02 H (test bench design time)	01 H (test bench design time)

From the Table 8.4 it is clear that the proposed PIVE framework outperformed system verilog based UVM.

8.2 RESULTS EXPLORATION ON REUSABILITY OF UNIVERSAL VERIFICATION METHODOLOGY

Figure 8.1 shows the inputs and outputs of RTL schemes of HUM framework.

Figure 8.1 Inputs and outputs of RTL schemes of the proposed HUM framework

After generating and creating the RTL schemes, various test inputs are given and accuracy/time is calculated for the different cases.

Table 8.5 Fault detection performance for the proposed framework

S.No	Type of Faults	Components in BLE	No of Faults Injected	No of Faults Detected
1	Bit Flip Faults	Correlator	200	200
2		HEC	200	198
3		Scrambler	200	200

Table 8.6 Comparative analysis of HUM and Exiting methodologies.

Features	HUM	UVM	AUM/OVM	BIST & ATPG
Reusability	Yes	Yes	No	No
Decision Phase	Intelligent	Manual	Manual	Manual
User friendly	Yes	No	No	No
Fault Diagnosis method	Yes	No	No	Yes
Accuracy in verification	Yes	Yes	No	No
Extendibility	Yes	Yes	No	No

From the Table 8.5 and Table 8.6 the proposed model shows its better performance. The proposed tool has been designed based on the Python based Direct programming and integration of the functional verification with the UVM features. The proposed tool offers several advantages such as the user friendly, integration of different tools in to single IDE, and High accuracy. The proposed tool can be further improvised by the implementation of the adaptive learning-based verification and storage which will make the system more robust.

8.4 RESULTS OF AN OPTIMIZED HYBRID BASED DEEP REINFORCEMENT NEURAL NETWORK FOR VERIFICATION METHODOLOGY

Table 8.7 depicts the coverage time for the different gate counts using the proposed framework. The proposed framework has shown notable performances in terms of coverage and time for the architecture, as mentioned above. The proposed framework's superiority is proved by comparing the framework with other existing frameworks such as UVM, HVF, HUM, and traditional CDF simulators [20,21].

Table 8.7 Computation time of the Proposed Framework for verifying the Different Gate Counts

S.No.	Number of Gates($x10^3$)	Time(s)
1	25	78.45
2	50	89.32
3	75	102.56
4	100	145.90
5	125	230.5
6	150	289.90
7	175	367.90
8	200	416.90

Figure 8.2 and Figure 8.3 represent the comparative analysis of coverage accuracy between the proposed framework and other existing frameworks [22,23]. The proposed framework has produced the best performance and maintains the highest coverage accuracy compared to other existing algorithms.

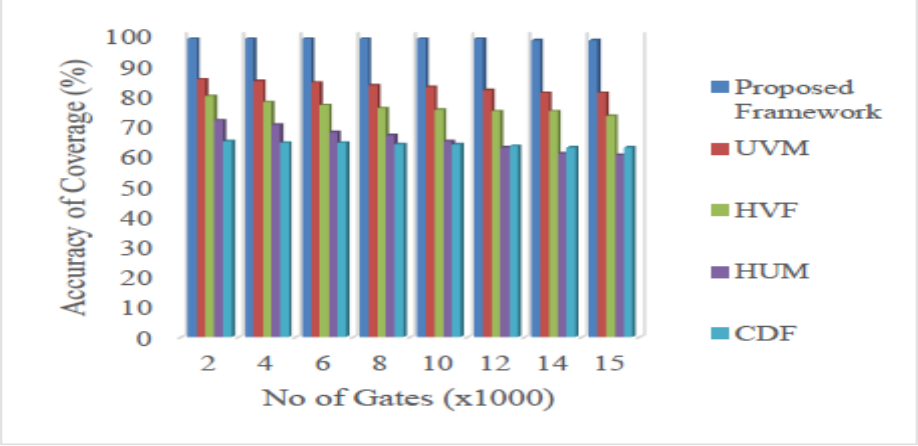

Figure 8.2 Comparative Analysis of accuracy of coverage in Bluetooth Baseband Architectures using the Different Intelligent Verification Framework

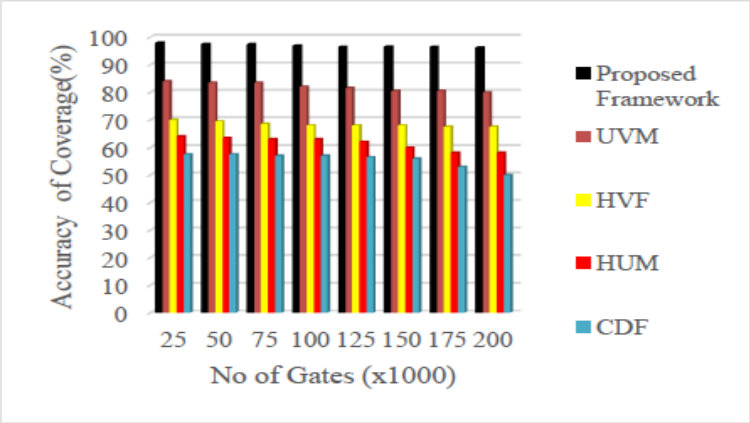

Figure 8.3 Comparative Analysis of accuracy of coverage in MIPS Processors using the Different Intelligent Verification Framework

As the number of input vectors increases, the proposed framework and UVM have produced a good performance. In contrast, the other existing algorithms have shown degraded performance as the input vectors increase, which is evident from Figures 8.2 and Figure 8.3 Though UVM has produced good results, the performance of the UVM is less as the input vectors increase.

Table 8.8 Speed of the Different Framework for verification of Different Architectures

S.No	Circuits Tested	Speed of the Algorithm				
		Proposed Framework	**UVM**	**HVF**	**HUM**	**CDF**
1	MIPS Architecture	4.5X	3X	2X	1.5X	0.7X
2	Bluetooth SoC	4.5X	3.5X	2X	1.8X	0.4X

Table 8.8 presents the speed of algorithms in verifying the different complex SoC architectures. It is found from the table 8.8, the proposed framework has better speed than the other existing verification mechanism.

Table 8.9 Overall performance Analysis between the different methods

No of Gates used	Methods-1 ARM based Mechanism	System Python Based Mechanism	DL based Mechanism	Remarks
25	127.90	92.34	78.45	Proposed DL based methods has outperformed the other model with the wide margin.
50	256.78	103.4	89.32	
75	549.0	143.5	102.56	
100	784.5	256.5	145.90	
125	1200.5	423.2	230.5	
150	1400.4	650.4	289.90	
175	2043.5	865.2	367.90	

The Table 8.9 infers that the proposed framework gives extraordinary performance and finds its application in verification methodology.

9. CONCLUSION

AVM has been proposed as the Integrated Scanning Environment which is implemented in ARM Architectures. We have proposed the new technique of verification methods which reduces the time of consumption in testing. The proposed AVM is enhanced version of UVM which integrates the API_commands for evoking the testing. Though AVM is hardware implementation of the function, still it needs improvisation for implementing for all SOC verification methodology.

The second phase of the research presents an outline of “python-based verification methodology” (PIVE). This tool significantly reduced the verification time when it is compared with system Verilog utilization for the UVM. The various contextual investigations plainly shows the decrease of time-frame regards of regression and development. Also, simple IP verification steps are provided and built highly efficient methodologies. Despite the fact that Python can be utilized in the check philosophy, mechanizing in discovering the bugs/shortcomings are absent.

In the third phase of the research UVM improves verification efficiency and reusability. This work provides a reusable I2C verification platform based on an analysis of the UVM standard verification environment and UVM technique to produce a reusable approach. The research shows that the proposed verification platform may be used to verify reusable I2C protocols.

The final phase of the research proposes a new UVM verification platform that integrates the RL and DELM. The proposed RL framework significantly learns the sequential environment of DUV and reaches the target vectors by providing the appropriate input sequences. The proposed framework verifies the combinational path once the target is attained. The experimental results show that the proposed RL and DELM framework outperforms the other algorithms in terms of no. of transitions. Also, a total speedup of 4.5x was confirmed compared to the other existing verification mechanism.

The proposed deep learning algorithm needs to be improvised to fit into the universal SoC verification mechanism as part of our future work.

REFERENCES

[1]. Khaled Salah, A UVM-Based Smart Functional Verification Platform: Concepts, Pros, Cons, and Opportunities, International Design and Test Symposium, (2014).

[2]. Alexander W. Rath, Volkan Esen and Wolfgang Ecker, A Transaction-Oriented UVM-Based Library for Verification of Analog Behaviour, Asia and South Pacific Design Automation Conference (ASP-DAC), (2014).

[3]. Ammon Gruwell, Peter Zabriskie, and Michael Wirthlin, High-Speed FPGA Configuration and Testing Through JTAG, *IEEE AUTOTESTCON*, (2016).

[4]. Khadija Tahera, David C. Wynn, Chris Earl and Claudia M. Eckert, Testing in the incremental design and development of complex products, Research in Engineering Design (30) (2019), 291–316

[5]. Mdhammadabdul Naveed and Dr.M.Srinivasan, An Effective Review of VLSI Design Performance Improvement In Computing Applications, Journal of critical Reviews **7** (12) (2020), 3625-3631.

[6]. Vithyalakshmi Natarajan, Low Power Design Methodology, *VLSI Intechope*n (2018), 73-729

[7]. K. Smirnov, A. Nazarov and V. Blinov, Methods of Automated Test Solutions Design for VLSI Testing, *International Conference on Industrial Engineering,Applications and Manufacturing (ICIEAM)*, (2020) pp. 1-6,

[8]. C. Xie, X.B.Li, Yu Lei and H.Y.Chen, BIST-Based Fault Diagnosis for PCM With Enhanced Test Scheme and Fault-Free Region Finding Algorithm, in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, **28** (7) (2020), pp. 1652-1664.

[9]. Varada, S., Katpally, S. and Thiruveedhi, S.S.L, Comprehensive Analysis and Optimization of Reliable Viterbi Decoder Circuits Implemented in Modular VLSI Design Logic Styles, *J Electron Test* (36) (2020), 343–363.

[10]. Uthayakumar, G O Jijina, G Suresh, and V Nagaraju, FPGA Based Approximate Digital VLSI Circuit Validating Focused on Fault Diagnosis, *Journal of Physics: Conference Series*, **1964** (6) (2021).

[11]. Shi X and Nicolici N, Generating cyclic-random sequences in a constrained space for in-system validation, *IEEE Transactions on Computers*. **65** (12) (2016), 3676–3686.

[12]. Gurha P and Khandelwal R, System verilog assertion based verification of AMBA-AHB, *ICMETE. IEEE*. **1** (74) (2016), 1694-7806

[13]. Chatterjee J, Saxena A, Mehra A, Vyas G and Mukesh V, Verification and debugging of lc-3 test bench environment using system Verilog, *ICECA, IEEE.* **2** (26) (2018), 1253–1258.

[14]. El Mandouh E and Wassal A G, Accelerating the debugging of fv traces using k-means clustering techniques, *International Design & Test Symposium* (IDT). IEEE. **11** (20) (2016), 278–283.

[15]. Park S U, Kim T P, Lee M Z, and Cho Y B, Method of RTL debugging when using HLS for HW design: Different simulation result of Verilog & VHDL, *ISOCC. IEEE*. **15** (12) (2018), 273–274.

[16]. S. Raghuvanshi and V. Singh, Review on Universal Verification Methodology (UVM) Concepts for Functional Verification, *International Journal of Electrical, Electronics and Data Communication,* **2** (3) (2014).

[17]. Ebrahimi, Abbas, Zandsalimy and Mohammad, Evaluation of FPGA Hardware as a New Approach for Accelerating the Numerical Solution of CFD Problems, *IEEE Access,* **5** (2017), 9717 - 9727.

[18]. Jian Wang, Siyuan Lu, Shui Hua Wang and Yu Dong Zhang, A review on extreme learning machine. *Multimedia Tools and Applications*, (81) (2021), 41611–41660

[19]. Hughes, William, Sandeep Srinivasan, Rohit Suvarna and Maithilee Kulkarni, Optimizing Design Verification using Machine Learning: Doing better than Random, *Computer science Bibliography* (2019), 1909-13168.

[20]. Suresh S and Ganesh R, FPGA Implementation of MIPS RISC Processor. *International Journal of Engineering Research & Technology*, **3** (1) (2014), 1711-1714.

[21]. Prasanth V, Sailaja V, Sunitha P and Vasantha Lakshmi B, Design and implementation of low power 5 stage Pipelined 32 bits MIPS Processor using 28nm Technology, *International Journal of Innovative Technology and Exploring Engineering*, **8** (42) (2020), 503-507.

[22]. Apoorva H M and Kiran Bailey, UVM based Design Verification of FIFO, *International Journal of Engineering Research & Technology,* **9** (6) (2020), 774- 776.

[23]. Gade S H, Sinha M, Rout S S, and Deb S, Enabling reliable high throughput on-chip wireless communication for many core architectures, *IEEE Computer Society Annual Symposium on VLSI (ISVLSI). IEEE*. (2018), 591–596.

LIST OF PUBLICATIONS

1. Bhuvaneswary, N., Deny, J., Lakshmi, A, Hybrid optimized verification methodology using deep reinforcement neural network, *Journal of Intelligent & Fuzzy Systems* (2023) (Accepted)

2. Bhuvaneswary, N., Deny, J., Lakshmi, A, Exploration on Reusability of Universal Verification Methodology, *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, IEEE, (2022)

3. Bhuvaneswary, N., Deny, J., Lakshmi, A, Design and implementation of intelligent ARM based verification methodology for low computation time and high performance, *International Journal of Advanced Science and Technology*, (28) (15) (2019).

4. Bhuvaneswary, N., Deny, J., Lakshmi, A, A Python Integrated Verification Environment for SoC Verification Methodologies in the Journal of Computing and Information Science in Engineering, (2023).